## Ejercicio 1

1. **Análisis del código:**

#!/bin/bash

#Este if testea que la cantidad de parámetros ingresados sea menor o igual a 2

if test $# -lt 2; then

echo "La cantidad de argumentos es menor a 2..."

exit 1;

fi

#Este if testea que el parámetro $1 sea un archivo válido

if ! test -e $1; then

echo "El argumento ($1) pasado por parámetro no es un archivo existente"

exit 1;

fi

FILE=$1 #Guarda el archivo ingresado por parámetro en la variable FILE

shift 1 #Desplaza en 1 posición el parámetro recibido, para posicionarme en el parámetro $2

COUNTER=0 #Setea en 0 el contador

while test $COUNTER -le $# ; do # Testea que la cantidad de palabras a buscar sea 2, o sea la cantidad de parámetros que permite ingresar el script

SET[$COUNTER]=$1 #Guarda la primera palabra ingresada por parámetro, para buscar en el archivo

shift 1 #Desplaza en 1 posición el parámetro recibido, para luego buscar la siguiente palabra ingresada por parámetro

((COUNTER++))

done

for X in "${SET[@]}" ;do

CANT=0

echo "contenido por parametro $X" #Aca busca el parámetro ingresado $2 en el archivo $1

for WORD in `cat $FILE` ;do #cat $FILE=Comando para visualizar el archivo $1 por pantalla

echo "contenido del archivo $WORD" #Lista todo el contenido del archivo dando una palabra por línea

if [[ "$WORD" == "$X" ]]; then

((CANT++)) #Pasa a la otra palabra, y vuelve a buscar y listar

fi

done

echo "$X : $CANT" #$X: Muestra la palabra buscada. $CANT: Muestra la cantidad de coincidencias en el archivo.

done

exit 0 #Fin del script

1. El script acepta 2 parámetros ingresados por línea de comandos. El primer parámetro es un archivo. El otro parámetro es una palabra a buscar en el mismo.
2. Recorre un archivo ingresado por parámetro ($1), listando línea por línea cada una de las palabras que lo conforman y busca las coincidencias del parámetro ($2) en ese archivo, listando la palabra buscada y cantidad de ocurrencias de la misma.

1. El objetivo general es buscar la cantidad de ocurrencias de una palabra en un archivo.
2. **shift**: La orden shift permite desplazar los parámetros de sitio, de tal forma que sean accesibles los que estén más allá del noveno, con el inconveniente de no poder acceder a los primeros.  
    **test:** El comando test evalúa una expresión, y si el resultado es verdadero, el estado de salida retornado es cero. En otro caso, si el resultado es falso, el estado de salida retornado es distinto de cero.  
    **cat:** Visualiza el contenido completo de un fichero de texto en la pantalla sin pausas.
3. **if:** El if permite testear una condición y en función del resultado cambiar el flujo de ejecución del script de la shell.   
     
   ***Formato del Comando***if [ expresion ]  
   then  
   *comando*else  
   *comando*fi
4. La evaluación de una expresión es la siguiente: Si el resultado es verdadero, el estado de salida retornado es cero. Si resulta falsa, se retorna un estado de salida distinto de cero.  
     
    for: Hay dos formas de utilizar un ciclo for en bash. Una de ellas es utilizando la estructura vieja de shell de Unix:  
   for VARIABLE in lista; do; N\*(comando;) done  
     
   Con esta sintaxis, la variable que figura en el encabezado, se va cargando con uno de los valores de la lista cada vez que entra a un ciclo del for.  
     
    La otra forma de utilizar un ciclo for en bash es muy parecida a como se utiliza el for en el lenguaje C o C++:  
     
   for (( *instrucción inicial; condición; instrucción de fin de cic*lo )); do;  
   N\*(comando;) done  
     
   while: El bucle while ejecuta un trozo de código si la expresión de control es verdadera, y sólo se para cuando es falsa (o se encuentra una interrupción explícita dentro del código en ejecución)  
     
   La sintaxis de esta construcción es la siguiente:  
   while *condición* do   
   *comandos*  
    done